ROBOPIANIST: Dexterous Piano Playing with Deep Reinforcement Learning

Anonymous Author(s)
Affiliation
Address
email

Abstract: Developing robotic hands capable of replicating human-like dexterity has been a grand challenge in robotics. While recent works have made impressive progress in the last few years with learning-based approaches, the studied tasks such as in-hand reorientation, relocation and simple tool use fail to capture the breadth of skills exhibited by human hands. To push the boundaries of high-dimensional control, we explore the task of controlling bi-manual anthropomorphic robotic hands to play the piano, which is targeted at testing high spatial and temporal precision, coordination, and planning, all with an underactuated system frequently making-and-breaking contacts. We propose a deep RL system, ROBOPIANIST, which can successfully learn a diverse repertoire of **150** piano pieces. The resulting policies exhibit highly dexterous behaviors, and produce visually and acoustically pleasing performances. Website featuring videos, code, and datasets is available at https://robopianist.github.io/.

Keywords: high-dimensional control, bi-manual dexterity

1 Introduction

Human hands built our world. Whether making a sharp stone tool, planting wheat, or typing on a keyboard, hands are humanity's essential effectors. While having some unique features—most famously, opposable thumbs—human hands are not significantly more mechanically capable than those of other primates. Instead, the differentiating factor is the human motor control system. The pursuit of replicating human-like dexterity in robotic hands has been a long-standing challenge in the field of robotics. Despite decades of research, achieving proficient control in high-dimensional robotic systems remains a grand challenge, inspiring a wealth of research from both mechanical design [1, 2, 3] and control theoretic perspectives [4, 5, 6, 7, 8]. Although learning-based approaches [9, 10, 11, 12, 13, 14, 15] have dominated recent literature, the class of problems typically considered corresponds to a narrow (when compared to humans) definition of dexterity. Most tasks are well-specified using a single goal-state or termination condition, limiting the complexity of the solution space and often yielding unnatural-looking behaviors that satisfy the terminal state. In order to bestow robots with artificial embodied intelligence that has the same precision, agility and dexterity of the human motor control system, we need to challenge our methods with similarly difficult tasks. In this work, we seek to push the limits of bi-manual dexterity through the challenging task of *piano playing* using simulated anthropomorphic robotic hands.

To this end, we propose ROBOPIANIST, a system that leverages deep reinforcement learning (RL) to synthesize policies capable of playing a diverse repertoire of 150 piano pieces. Playing the piano requires high spatial and temporal precision, coordination, and planning, all with an underactuated system that frequently makes-and-breaks contacts. Furthermore, success is graded on accuracy over an entire episode, concretely, via a time-varying non-analytic output of the environment, i.e., the music. Additionally, piano playing presents an immediately *interpretable* task success signal

Submitted to the 7th Conference on Robot Learning (CoRL 2023). Do not distribute.

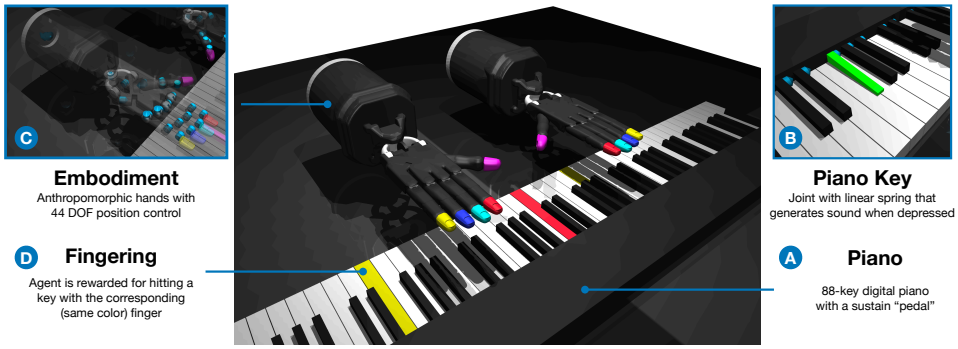


Figure 1: **ROBOPIANIST** simulation featuring a full-size digital keyboard (A) with 88 piano keys modeled as linear springs (B). In the piano playing task, two (left and right) anthropomorphic Shadow hands (C) are tasked with playing a musical piece encoded as a trajectory of key presses (D).

(i.e., "does it sound good?"), which can be evaluated independently from the particular reward/cost functions used during training.

Our contributions are as follows: **(i)** We introduce ROBOPIANIST, a system that enables two anthropomorphic robotic hands to learn a diverse repertoire of piano pieces using deep RL, exhibiting remarkable dexterity and producing visually and acoustically pleasing performances, **(ii)** We provide an open source benchmark environment with 150 songs, which we title ROBOPIANIST-REPERTOIRE-150, allowing for a comprehensive evaluation of the proposed system and providing a challenging high dimensional control environment for future research, **(iii)** We demonstrate the effectiveness of multi-task imitation learning in training a single policy capable of playing multiple songs, **(iv)** We analyze and ablate various system design considerations, providing insights for future high-dimensional control settings.

2 Related Work

We address related work within two primary areas: dexterous high-dimensional control, and robotic pianists. For a more comprehensive related work, please see Appendix A.

Dexterous Manipulation as a High-Dimensional Control Problem The vast majority of the manipulation literature uses lower-dimensional systems (i.e., single-arm, simple end-effectors), which circumvents challenges that arise in more complex systems. Specifically, only a handful of general-purpose policy optimization methods have been shown to work on high-dimensional hands, even for a single hand [10, 9, 12, 11, 16, 14, 17, 7], and of these, only a subset has demonstrated results in the real world [10, 9, 12, 11, 16]. Results with bi-manual hands are even rarer [15, 18]. In addition, the class of problems generally tackled in these settings corresponds to a definition of dexterity pertaining to traditional manipulation skills [19], such as re-orientation, relocation, manipulating simply-articulated objects (e.g., door-opening, ball throwing and catching), and using simple tools (e.g., hammer) [20, 21, 15, 11, 22, 12]. This effectively reduces the search space for controls to predominantly a single "basin-of-attraction" in behavior space per task. In contrast, our suite of tasks encompasses a more complex notion of a goal, which is encoded through a musical piece.

Robotic Piano Playing Robotic pianists have a rich history within the literature, with several works dedicated to the design of specialized hardware [23, 24, 25, 26, 27, 28], and/or customized controllers for playing back a song using pre-programmed commands [29, 30]. The works of Scholz [31], Yeon [32] use a dexterous hand to play the piano by leveraging a combination of inverse kinematics and offline trajectory planning. In Xu et al. [33], the authors formulate piano playing as an RL problem for a single Allegro hand on a miniature piano and leverage tactile sensor feedback. The piano playing tasks considered in these prior works are relatively simple (e.g., play up to six successive notes, or three successive chords with only two simultaneous keys pressed for each chord). On the other hand, ROBOPIANIST allows a general bi-manual controllable agent to emulate a pianist's growing proficiency by providing a large range of musical pieces with graded difficulties.

3 Experimental Setup

Our goal is to study highly complex, dexterous bi-manual control with an eye toward achieving human-like robotic capabilities. To this end, we introduce a piano-playing environment with a challenging curriculum of tasks, where each task is a separate musical piece to play. We detail the design of the environment as well as the musical suite we use to train and evaluate our agent.

Simulation details. We build our simulated piano-playing environment (depicted in Figure 1) using the open-source MuJoCo [34] physics engine with the `dm_control` Python bindings [35]. The piano model is a full-size digital keyboard with 52 white keys and 36 black keys, using a Kawai manual [36] as reference for the keys' positioning and dimensions. The left and right hands are Shadow Dexterous Hand [37] models from MuJoCo Menagerie [38], which have been designed to closely reproduce the kinematics of the human hand. We use the Musical Instrument Digital Interface (MIDI) standard to represent musical pieces and to synthesize sound from key presses. In particular, each key is modeled as a joint with a linear spring and is considered 'active' when its joint position is within 0.5 degrees of its maximum range, at which point a synthesizer is used to generate its corresponding note sound. We also implement a mechanism to sustain the sound of any currently active note to mimic the mechanical effect of a sustain pedal in real pianos.

Musical representation. We use the Musical Instrument Digital Interface (MIDI) standard to represent musical pieces and synthesize sounds. A given song's MIDI file is converted into a piano roll, i.e., a binary matrix in $\mathbb{Z}^{T\times 88}$, where T is the number of time steps (a function of the MIDI length and the robot control frequency). The piano roll is a time-indexed trajectory telling us which of the 88 notes should be active at every time step, essentially defining a time-varying "goal trajectory."

MDP formulation. We model piano-playing as a finite-horizon Markov Decision Process (MDP) defined by a tuple $(\mathcal{S}, \mathcal{A}, \rho, p, r, \gamma, H)$ where $\mathcal{S} \subset \mathbb{R}^n$ is the state space, $\mathcal{A} \subset \mathbb{R}^m$ is the action space, $\rho(\cdot)$ is the initial state distribution, $p(\cdot|s,a)$ governs the dynamics, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ defines the rewards, $\gamma \in [0,1)$ is the discount factor, and H is the horizon. The goal of an agent is to maximize its total expected discounted reward over the horizon $\mathbb{E}\left[\sum_{t=0}^{H} \gamma^t r(s_t, a_t)\right]$.

The agent observes proprioceptive (i.e., hand joint positions) and exteroceptive (i.e., piano key joint positions) state information. Although in principle, this is all the information required for the task as defined, we found that current methods are unable to cope with the exploration challenge (see Section 4). To provide some form of supervision, we include fingering information in the form of a boolean vector indicating which fingers should be in contact with the piano at that time. The agent must be aware of at least a few seconds' worth of its next goals in order to be able to plan appropriately. So at each time step, the agent takes as input the song's piano roll for some lookahead horizon. A detailed description of the observation space is given in Table 1. We define actions to be target joint angles for the 22 joints in each hand and one for the sustain pedal (for a total of 45 dimensions) that are taken at 20Hz. These targets are fed to proportional-position actuators which convert them to torques at the 52 hand joints. The objective is to press the goal keys (and *only* the goal keys) at each time step for T steps, where T is the number of time steps needed to play the prescribed song. Since the reward function is crucial for learning performance, we discuss its design in Section 4, and the exact, detailed reward formulation can be found in Appendix B.

Observations	**Unit**	**Size**
Hand and forearm joints	rad	52
Forearm Cartesian position	m	6
Piano key joints	rad	88
Fingering	discrete	$L \cdot 10$
Piano key goal state	discrete	$L \cdot 88$

Table 1: The details of the agent's observation space. L corresponds to the lookahead horizon.

Evaluation criteria. We introduce precision, recall, and F1 scores as interpretable evaluation metrics on top of total return achieved. These metrics are computed by comparing the state of the piano keys at every time step with the corresponding ground-truth state stored in the MIDI file, averaged across all time steps. Intuitively, precision measures how good a policy is at not hitting keys that

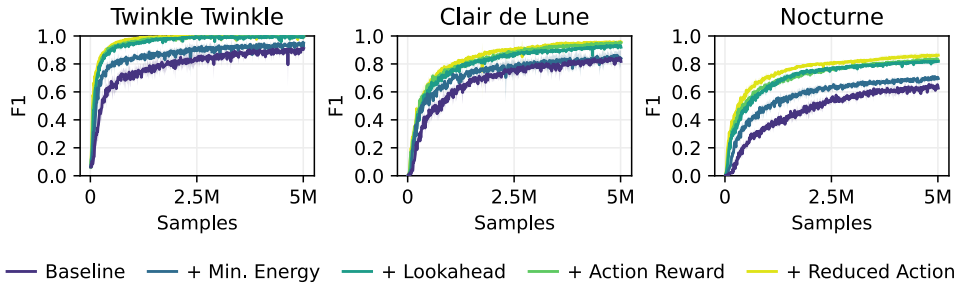


Figure 2: **RoboPianist Design Considerations**. The F1 performance for 3 songs of varying difficulties. From left to right, in the order specified by the legend, each curve **inherits all MDP attributes of the curve before it** but makes the additional modification as specified by its label. We see that each addition is beneficial in improving F1 performance. In particular, the additions of the energy penalty and lookahead on the goal sequence are critically important. We note that reducing the action space simplifies learning, as is seen most distinctly in the hardest of the 3 pieces: Nocturne.

should not be played (i.e., low false positive rate), and recall measures how good it is at hitting the keys that should be played (i.e., low false negative). For example, a policy that hovers above the keyboard without hitting any keys has high precision but low recall; conversely, a policy that hits all the keys simultaneously has high recall but low precision. The F1 score combines the precision and recall into a single metric using their harmonic mean and ranges from 0 (if either precision or recall is 0) to 1 (perfect precision and recall). We primarily use the F1 score when evaluating as it is a common heuristic accuracy score for various audio information retrieval or signal processing tasks [39] and we found empirically that it correlates with qualitative performance on our tasks.

Task suite. One of our main contributions is a diverse set of piano-playing tasks with which we can evaluate high-dimensional control. We take the PIG dataset [40], developed for research into estimating human fingering for piano, which contains piano pieces from 24 Western composers spanning baroque, classical and romantic periods that vary in difficulty, ranging from relatively easy (e.g., Mozart's Piano Sonata K 545 in C major) to significantly harder (e.g., Scriabin's Piano Sonata No. 5) and range in length from tens of seconds to over 3 minutes, and transform it into a corpus of 150 MIDI files that can be played in our simulated environment with fingering annotations.

4 RoboPianist System Design

Our aim is to enable robots to exhibit sophisticated, high-dimensional control necessary for successfully performing challenging musical pieces. Learning the task requires (i) spatial and temporal precision (hitting the right notes, at the right time), (ii) coordination (simultaneously achieving multiple different goals, in this case, fingers on each hand hitting different notes, without colliding), and (iii) planning (how a key is pressed should be conditioned on the expectation of how it would enable the policy to reach future notes). These behaviors do not emerge if we solely optimize with a sparse reward for pressing the right keys at the right times. The main challenge is exploration, which is further exacerbated by the high-dimensional control space. We overcome this challenge by leveraging human priors and careful system design, which we detail below.

Human priors. Piano fingering refers to the assignment of fingers to notes to be played for a particular passage that sets up the pianist in a position to be able to play the oncoming notes naturally and efficiently. In order to guide a pianist towards the most efficient way to play a musical piece, sheet music will typically provide sparse fingering information, usually reserved for particularly tricky sections of a piece. Pianists rely on their expertise and personal experience to discover fingerings for the rest of the piece. Further, most fingering markings on sheet music are a general guide, and it is up to the pianist to gracefully incorporate this assistive information and ensure a smooth musical flow. For RoboPianist, we found that the prior of fingering information was critical for learning. In our presented solution methodologies, we incorporate the boolean fingering information within

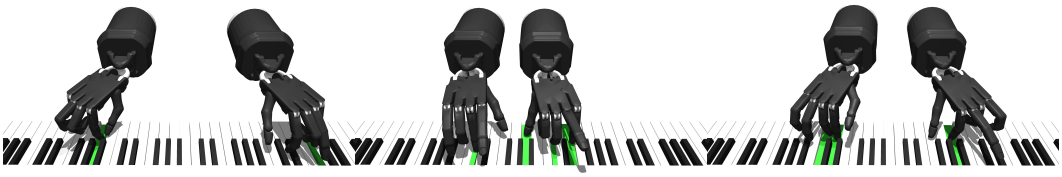


Figure 4: Policies in the ROBOPIANIST environment displaying skilled piano behaviors such as (1) simultaneously controlling both hands to reach for notes on opposite ends of the keyboard, (2) playing a chord with the left hand, which involves precisely and simultaneously hitting a note triplet, and (3) playing a trill with the right hand, which involves rapidly alternating between two adjacent notes.

the reward formulation (Table 2). When we remove this prior and only reward the agent for the key press, the agent's F1 stays at zero and no substantial learning progress is made. We suspect that the benefit of fingering comes not only from helping the agent achieve the current goal, but facilitating key presses in subsequent timesteps. Having the policy discover its own preferred fingering, like an experienced pianist, is an exciting direction for future research.

Reward design. The reward at a given time step is a weighted sum over the following terms. We first include a reward proportional to how depressed the keys that should be active are. We then add a constant penalty if any inactive keys are pressed hard enough to produce sound. This gives the agent some leeway to rest its fingers on inactive keys so long as they don't produce sound. We found that giving a constant penalty regardless of the number of false positives was crucial for learning; otherwise, the agent would become too conservative and hover above the keyboard. In contrast, the smooth reward for pressing active keys plays an important role in exploration by providing a dense learning signal. We introduce two additional shaping terms: (i) we encourage the fingers to be spatially close to the keys they need to press (as prescribed by the fingering labels) to help in exploration and (ii) we penalize energy, which reduces variance and erratic behavior that high-dimensional control policies trained with RL are prone to.

Looking into the future leads to more temporally precise control. We observe additional improvements in performance and variance from including future goal states in the observation. Intuitively, this allows the policy to plan better for future notes – for example by placing the non-finger joints (e.g., the wrist) in a manner that allows more timely reaching of notes at the next timestep.

Constraining the action space trains faster and reduces un-human-like movement. To alleviate exploration even further, we explore disabling degrees of freedom in the Shadow Hand that either do not exist in the human hand (e.g., the little finger being opposable) or are not strictly necessary for most songs. We additionally reduce the joint range of the thumb, which on the Shadow Hand, can reach backward in a manner impossible for humans. Adding these constraints speeds up learning considerably but limits its potential capabilities. As seen in Figure 2, training with the full action space catches up to the reduced action space with additional training time.

5 Results

In this section, we present our experimental findings on ROBOPIANIST-ETUDE-12, a subset of ROBOPIANIST-REPERTOIRE-150, for clarity—full results can be found in Appendix C. We design our experiments to answer the following questions:

(1) How does our method compare to a strong baseline in being able to play individual pieces?
(2) How can we enable a single policy learn to play more than one song?
(3) What effects do our design decisions have on the feasibility of acquiring highly complex, dexterous control policies?

5.1 Specialist Policy Learning

For our policy optimizer, we use a state-of-the-art model-free RL algorithm DroQ [41]. We evaluate MPC, which leverages the simulator as a model, as a meaningful point of comparison. We use the implementation from Howell et al. [42] which was shown to solve previously-considered-challenging dexterous manipulation tasks [9, 13, 11] in simulation. Amongst various options [42], we found the most success with "Predictive Sampling", a *derivative-free* sampling-based method. A detailed discussion of this choice of baseline and its implementation can be found in Appendix F. Our method uses 5 million samples to train for each song using the same set of hyperparameters, and the MPC method is run at one-tenth of real-time speed to give the planner adequate search time.

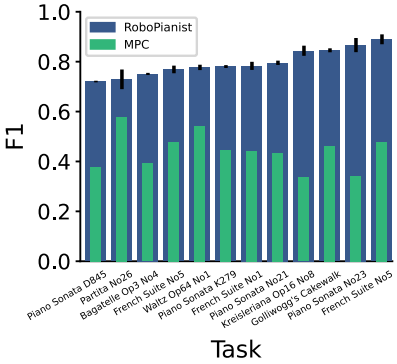


Figure 3: The F1 scores achieved by ROBOPIANIST (blue) and MPC (green) for each of the ROBOPIANIST-ETUDE-12 tasks.

The quantitative results are shown in Figure 3. We observe that the ROBOPIANIST agent significantly outperforms the MPC baseline, achieving an average F1 score of 0.79 compared to 0.43 for MPC. We hypothesise that the main bottleneck for Predictive Sampling is compute: the planner struggles with the large search space which means the quality of the solutions suffer in the fixed time budget. Qualitatively, our learned policies display remarkably skilled piano behaviors such as (1) simultaneously controlling both hands to reach for notes on opposite ends of the keyboard, (2) playing *chords* which involves precisely and simultaneously hitting note triplets, and (3) playing a *trill* with the right hand, which involves rapidly alternating between two adjacent notes (see Figure 4). We encourage the reader to listen to these policies on the supplementary website[1].

5.2 Multi-Song Policy Learning

Multitask RL Ideally, a single model would be able to learn how to play all songs. Secondly, we would like the policy to generalize to new songs. To facilitate this, we set up a multi-task environment where each song is a new task. When the environment is reset in this training regime, we randomly sample a song from the list of available songs. In our multi-song setting, each successive environment is additive (i.e., the 2-song environment contains the same song as the 1-song environment, plus a new one). We first test whether we can have a single model play multiple songs by training with various multi-song environments, where the 1-song environment involves learning only Für Elise, and analyzing the resulting models' performance on this shared song. We report results of applying ROBOPIANIST adapted directly to this setting in Figure 5 and see that training on multiple songs makes it significantly harder to learn. As we increase the songs the agent is tasked with learning, the F1 score on the shared training piece greatly decreases. We also test generalization capabilities by evaluating the same models' performance when evaluated without fine-tuning on ROBOPIANIST-ETUDE-12 (which does not have overlap with the training songs). We see that, perhaps surprisingly, multitask RL training does not provide any benefit in generalization, where the learned policy cannot be used zero-shot on a novel song.

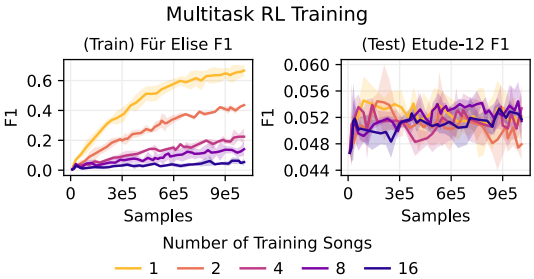
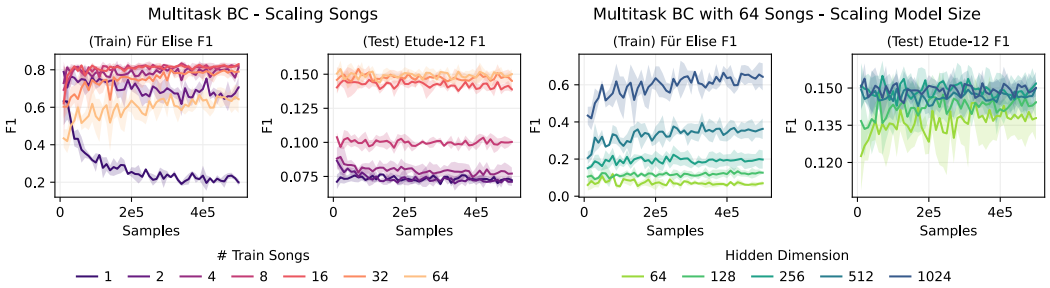


Figure 5: **Multi-song training with RL**. We test the ability to learn multiple songs simultaneously. The training performance on the shared song, Für Elise, is shown on the left. We see adding more songs degrades performance on individual songs. On the right, we evaluate on the held out ROBOPIANIST-ETUDE-12 songs, suggesting the difficulty of generalization with multi-task RL training.

[1] https://robopianist.github.io/



(a) Scaling songs for a model with a hidden dimension of 1024.

(b) Scaling the model size trained on 64 songs.

Figure 6: Here we show the learning curves of BC, showing the interplay between the attempted number of training songs and model size. The song Für Elise, is in the training dataset for each experiment. We show the F1 score across Für Elise and ROBOPIANIST-ETUDE-12. We see that for a small number of songs, the model quickly overfits, suggesting that for the given model size the amount of data and diversity is lacking. We see that adding more songs results in a higher test F1, suggesting that the model is beginning to generalize. However, zero-shot test performance is far from that of the expert.

Multitask Behavioral Cloning Since multitask training with RL proves challenging with the single-song approach, we acquire a multitask policy by distilling specialist RL policies with behavioral cloning (BC) [43]. We collect 100 trajectories for each song in the ROBOPIANIST-REPERTOIRE-150 and hold out 2 trajectories for validation, and use the trajectory data of the ROBOPIANIST-ETUDE-12 songs for testing. We train a neural network to predict the expert's actions given the current state and goal using a standard mean squared error loss. Our multi song dataset uses the same song progression as the multitask RL environments, allowing a more direct comparison. Results are shown in Figures 6a. Unlike RL, we observe that as we increase the number of songs in the training dataset, the model's ability to generalize improves, resulting in higher test F1 scores. We note that for a large model, training on too few songs results in overfitting, which can be attributed to not having enough data. Using a smaller model alleviates this issue, as shown in the more detailed multitask BC results found in Appendix E, but smaller models are unable to perform well on multiple songs. Despite having better generalization performance than RL, zero-shot performance on the ROBOPIANIST-ETUDE-12 falls far below the performance achived by a single-task expert. Additionally, we investigate the effect of model size on the multitask BC performance. We train BC models with different hidden dimension sizes on a dataset with 64 songs. As shown in Figure 6b, a smaller hidden dimension size result in lower F1 performance, as the model has difficulty fitting the songs. As we increase the model size, the F1 score on the training song improves, suggesting that larger models can better capture the complexity of the piano playing task across multiple songs.

5.3 Further analysis

We built and explored a wide range of features in the ROBOPIANIST environment, as described in Section 4. The effects of an energy penalty, future lookahead on the goals, adding action and reward to the environment, and reducing the action space are shown in Figure 2 and analyzed below.

Control frequency and lookahead horizon: Figure 7 illustrates the interplay between control timestep, defined as the reciprocal of control frequency, and lookahead time in relation to their combined effect on F1 performance. A control timestep that is too large can result make it impossible to play faster songs, while a smaller control timestep increases the effective horizon the agent has to consider, thereby increasing the computational complexity of the task. Lookahead on the other hand specifies the number of future control steps for which goal states are provided. As the control timestep decreases, the lookahead must be increased to maintain the same amount of real-world time the agent sees about its future goals. These two parameters jointly modulate the control granularity and foresight of the agent. We find that a control frequency of 20Hz (0.05 seconds) is a sweet spot, with notably 100 Hz (0.01 seconds) drastically reducing the final score. At 100Hz, the MDP

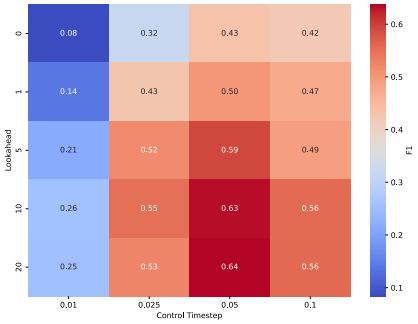


Figure 7: **Lookahead & Control Timestep**. These parameters jointly modulate the foresight and control granularity of the agent. We find a control timestep of 0.05 offers optimal control granularity, and performance generally improves with the lookahead up to a point.

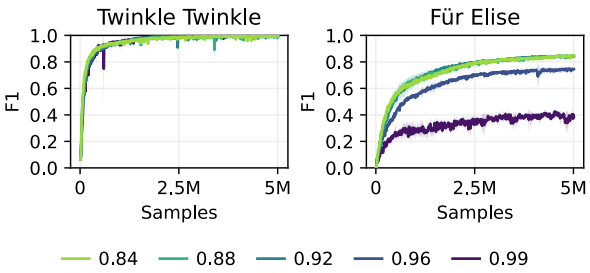


Figure 8: **Discount factor**. We find the discount factor to have a high impact on F1, especially as the task complexity increases. While on simpler songs (Twinkle Twinkle) the discount factor does not have any visible effect, it is critical for faster-paced songs (Für Elise). The curves show that any discount in the range of 0.84-0.92 can be use with similar effect.

becomes too long-horizon, which complicates exploration, and at 10Hz, the discretization of the MIDI file becomes too coarse, which negatively impacts the timing of the notes.

Discount factor: As shown in Figure 8, a proper discount factor is very important for achieving high F1 performance. We sweep over a range of discount factors on Twinkle Twinkle, which is a simple song, and Für Elise, which has a faster tempo and is a more difficult song. Notably, we find that using a discount factor of 0.99, as is commonly used in the RL literature, results in much lower performance than alternative values. We find that any discount around the range of 0.84 and 0.92 can produce similarly performant policies. Too high a discount factor such as 0.99 or 0.96 results in lower F1 performance. Qualitatively on Für Elise, we noticed that agents trained with the higher discount of 0.96 and 0.99 often opts to skip entire sub-segments of notes. We suspect that this is due to the long horizon and complex nature of the piano playing task, where the return is a function of the future goals vectors. A high discount factor makes it difficult for the critic to fit as it cannot reason about rewards past the provided lookahead time.

6 Discussion

Limitations While ROBOPIANIST produces agents that push the boundaries of bi-manual dexterous control, it does so in a simplified simulation of the real world. For example, we do not use the velocity of the key presses present in the MIDI file, which means our agent does not respect the dynamic markings of the composition. This is not the case in the real world. Furthermore, our RL training approach can be considered wasteful, in that we learn by attempting to play the entire piece at the start of every episode, rather than focusing on the parts of the song that have low F1. Finally, our results highlight that multitask learning is challenging, especially in the RL setting.

Conclusion In this paper, we introduced ROBOPIANIST, which provides a simulation framework and suite of tasks in the form of a corpus of songs, together with a high-quality baseline and various axes of evaluation, for studying the challenging high-dimensional control problem of mastering piano-playing with two hands. Our results demonstrate the effectiveness of our approach in learning a broad repertoire of musical pieces, and highlight the importance of various design choices required for achieving this performance. There is an array of exciting future directions to explore with ROBOPIANIST including: pushing the boundaries of both model-based and model-free methods, leveraging human priors to accelerate learning (e.g., motion priors from YouTube), studying zero-shot generalization to new songs, and incorporating multimodal data such as sound and touch. We believe that ROBOPIANIST serves as a valuable platform for the research community, enabling further advancements in high-dimensional control and dexterous manipulation.

References

[1] S. Yuan, L. Shao, C. L. Yako, A. M. Gruebele, and J. K. Salisbury. Design and control of roller grasper v2 for in-hand manipulation. *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 9151–9158, 2020.

[2] Z. Xu and E. Todorov. Design of a highly biomimetic anthropomorphic robotic hand towards artificial limb regeneration. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3485–3492, 2016. doi:10.1109/ICRA.2016.7487528.

[3] C. McCann, V. Patel, and A. Dollar. The stewart hand: A highly dexterous, six-degrees-of-freedom manipulator based on the stewart-gough platform. *icra*, 28(2):23–36, 2021. doi:10.1109/MRA.2021.3064750.

[4] R. Fearing. Implementing a force strategy for object re-orientation. In *Proceedings. 1986 IEEE International Conference on Robotics and Automation*, volume 3, pages 96–102, 1986. doi:10.1109/ROBOT.1986.1087655.

[5] D. Rus. In-hand dexterous manipulation of piecewise-smooth 3-d objects. *The International Journal of Robotics Research*, 18(4):355–381, 1999.

[6] A. Okamura, N. Smaby, and M. Cutkosky. An overview of dexterous manipulation. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, volume 1, pages 255–262 vol.1, 2000. doi:10.1109/ROBOT.2000.844067.

[7] T. Pang, H. J. T. Suh, L. Yang, and R. Tedrake. Global planning for contact-rich manipulation via local smoothing of quasi-dynamic contact models. *ArXiv*, abs/2206.10787, 2022.

[8] R. R. Ma and A. M. Dollar. On dexterity and dexterous manipulation. In *2011 15th International Conference on Advanced Robotics (ICAR)*, pages 1–7, 2011. doi:10.1109/ICAR.2011.6088576.

[9] M. Andrychowicz, B. Baker, M. Chociej, R. Józefowicz, B. McGrew, J. W. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, J. Schneider, S. Sidor, J. Tobin, P. Welinder, L. Weng, and W. Zaremba. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39:20 – 3, 2018.

[10] OpenAI, I. Akkaya, M. Andrychowicz, M. Chociej, M. Litwin, B. McGrew, A. Petron, A. Paino, M. Plappert, G. Powell, R. Ribas, J. Schneider, N. A. Tezak, J. Tworek, P. Welinder, L. Weng, Q. Yuan, W. Zaremba, and L. M. Zhang. Solving rubik's cube with a robot hand. *ArXiv*, abs/1910.07113, 2019.

[11] A. Handa, A. Allshire, V. Makoviychuk, A. Petrenko, R. Singh, J. Liu, D. Makoviichuk, K. Van Wyk, A. Zhurkevich, B. Sundaralingam, et al. Dextreme: Transfer of agile in-hand manipulation from simulation to reality. *arXiv preprint arXiv:2210.13702*, 2022.

[12] T. Chen, J. Xu, and P. Agrawal. A system for general in-hand object re-orientation. In *Conference on Robot Learning*, pages 297–307, 2022.

[13] A. Nagabandi, K. Konolige, S. Levine, and V. Kumar. Deep dynamics models for learning dexterous manipulation. *Conference on Robot Learning (CoRL)*, abs/1909.11652, 2019.

[14] T. Chen, M. Tippur, S. Wu, V. Kumar, E. H. Adelson, and P. Agrawal. Visual dexterity: In-hand dexterous manipulation from depth. *ArXiv*, abs/2211.11744, 2022.

[15] Y. Chen, Y. Yang, T. Wu, S. Wang, X. Feng, J. Jiang, S. M. McAleer, H. Dong, Z. Lu, and S.-C. Zhu. Towards human-level bimanual dexterous manipulation with reinforcement learning. *arXiv preprint arXiv:2206.08686*, 2022.

[16] H. Qi, A. Kumar, R. Calandra, Y. Ma, and J. Malik. In-Hand Object Rotation via Rapid Motor Adaptation. In *Conference on Robot Learning (CoRL)*, 2022.

[17] I. Mordatch, E. Todorov, and Z. Popović. Discovery of complex behaviors through contact-invariant optimization. *ACM Trans. Graph.*, 31(4), jul 2012. ISSN 0730-0301. doi:10.1145/2185520.2185539. URL https://doi.org/10.1145/2185520.2185539.

[18] A. M. Castro, F. N. Permenter, and X. Han. An unconstrained convex formulation of compliant contact. *IEEE Transactions on Robotics*, 2022.

[19] R. R. Ma and A. M. Dollar. On dexterity and dexterous manipulation. In *2011 15th International Conference on Advanced Robotics (ICAR)*, pages 1–7. IEEE, 2011.

[20] J. Fu, A. Kumar, O. Nachum, G. Tucker, and S. Levine. D4RL: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

[21] C. Smith, Y. Karayiannidis, L. Nalpantidis, X. Gratal, P. Qi, D. V. Dimarogonas, and D. Kragic. Dual arm manipulation—a survey. *Robotics and Autonomous systems*, 60(10):1340–1353, 2012.

[22] H. J. Charlesworth and G. Montana. Solving challenging dexterous manipulation tasks with trajectory optimisation and reinforcement learning. In *International Conference on Machine Learning*, pages 1496–1506. PMLR, 2021.

[23] I. Kato, S. Ohteru, K. Shirai, T. Matsushima, S. Narita, S. Sugano, T. Kobayashi, and E. Fujisawa. The robot musician 'wabot-2'(waseda robot-2). *Robotics*, 3(2):143–155, 1987.

[24] J.-C. Lin, H.-H. Huang, Y.-F. Li, J.-C. Tai, and L.-W. Liu. Electronic piano playing robot. In *2010 International Symposium on Computer, Communication, Control and Automation (3CA)*, volume 2, pages 353–356. IEEE, 2010.

[25] T. Maloney. Piano-playing robotic arm. Technical report, 2019.

[26] J. Hughes and P. Maiolino. An anthropomorphic soft skeleton hand exploiting conditional models for piano playing. *Science Robotics*, 3(25), 2018.

[27] R. Castro Ornelas. *Robotic Finger Hardware and Controls Design for Dynamic Piano Playing*. PhD thesis, Massachusetts Institute of Technology, 2022.

[28] D. Zhang, J. Lei, B. Li, D. Lau, and C. Cameron. Design and analysis of a piano playing robot. In *2009 International Conference on Information and Automation*, pages 757–761, 2009. doi:10.1109/ICINFA.2009.5205022.

[29] Y.-F. Li and L.-L. Chuang. Controller design for music playing robot—applied to the anthropomorphic piano robot. In *2013 IEEE 10th International Conference on Power Electronics and Drive Systems (PEDS)*, pages 968–973. IEEE, 2013.

[30] A. Zhang, M. Malhotra, and Y. Matsuoka. Musical piano performance by the act hand. In *2011 IEEE international conference on robotics and automation*, pages 3536–3541. IEEE, 2011.

[31] B. Scholz. *Playing Piano with a Shadow Dexterous Hand*. PhD thesis, Universität Hamburg, 2019.

[32] S. Yeon. Playing piano with a robotic hand, 2022. URL https://github.com/seongho-yeon/playing-piano-with-a-robotic-hand.

[33] H. Xu, Y. Luo, S. Wang, T. Darrell, and R. Calandra. Towards learning to play piano with dexterous hands and touch. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 10410–10416. IEEE, 2022.

[34] E. Todorov, T. Erez, and Y. Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012. doi:10.1109/IROS.2012.6386109.

[35] S. Tunyasuvunakool, A. Muldal, Y. Doron, S. Liu, S. Bohez, J. Merel, T. Erez, T. Lillicrap, N. Heess, and Y. Tassa. dm_control: Software and tasks for continuous control. *Software Impacts*, 6:100022, 2020. ISSN 2665-9638. doi:https://doi.org/10.1016/j.simpa.2020.100022. URL https://www.sciencedirect.com/science/article/pii/S2665963820300099.

[36] K. M. Instruments. Kawai Vertical Piano Regulation Manual, 2019. URL https://kawaius.com/wp-content/uploads/2019/04/Kawai-Upright-Piano-Regulation-Manual.pdf.

[37] Shadow Dexterous Hand, 2005. URL https://www.shadowrobot.com/dexterous-hand-series/.

[38] M. M. Contributors. MuJoCo Menagerie: A collection of high-quality simulation models for MuJoCo, 2022. URL http://github.com/deepmind/mujoco_menagerie.

[39] C. Raffel, B. McFee, E. J. Humphrey, J. Salamon, O. Nieto, D. Liang, and D. P. W. Ellis. Mir_eval: A transparent implementation of common mir metrics. In *International Society for Music Information Retrieval Conference*, 2014.

[40] E. Nakamura, Y. Saito, and K. Yoshii. Statistical learning and estimation of piano fingering. *ArXiv*, abs/1904.10237, 2019.

[41] T. Hiraoka, T. Imagawa, T. Hashimoto, T. Onishi, and Y. Tsuruoka. Dropout q-functions for doubly efficient reinforcement learning. *International Conference on Learning Representations (ICLR)*, 2022.

[42] T. A. Howell, N. Gileadi, S. Tunyasuvunakool, K. Zakka, T. Erez, and Y. Tassa. Predictive sampling: Real-time behaviour synthesis with mujoco. *ArXiv*, abs/2212.00541, 2022.

[43] D. A. Pomerleau. Alvinn: An autonomous land vehicle in a neural network. In *NIPS*, 1988.

[44] A. M. Okamura, N. Smaby, and M. R. Cutkosky. An overview of dexterous manipulation. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, volume 1, pages 255–262. IEEE, 2000.

[45] A. Rajeswaran, V. Kumar, A. Gupta, J. Schulman, E. Todorov, and S. Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. *ArXiv*, abs/1709.10087, 2017.

[46] B. Sundaralingam and T. Hermans. Relaxed-rigidity constraints: kinematic trajectory optimization and collision avoidance for in-grasp manipulation. *Autonomous Robots*, 43:469–483, 2019.

[47] I. Radosavovic, X. Wang, L. Pinto, and J. Malik. State-only imitation learning for dexterous manipulation. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7865–7871. IEEE, 2021.

[48] N. Heess, D. TB, S. Sriram, J. Lemmon, J. Merel, G. Wayne, Y. Tassa, T. Erez, Z. Wang, S. Eslami, et al. Emergence of locomotion behaviours in rich environments. *arXiv preprint arXiv:1707.02286*, 2017.

[49] A. Sharma, S. Gu, S. Levine, V. Kumar, and K. Hausman. Dynamics-aware unsupervised discovery of skills. In *International Conference on Learning Representations*, 2020.

[50] S. Tunyasuvunakool, A. Muldal, Y. Doron, S. Liu, S. Bohez, J. Merel, T. Erez, T. Lillicrap, N. Heess, and Y. Tassa. dm_control: Software and tasks for continuous control. *Software Impacts*, 6:100022, 2020.

[51] X. B. Peng, Z. Ma, P. Abbeel, S. Levine, and A. Kanazawa. AMP: Adversarial motion priors for stylized physics-based character control. *ACM Transactions on Graphics (TOG)*, 40(4): 1–20, 2021.

[52] J. Merel, L. Hasenclever, A. Galashov, A. Ahuja, V. Pham, G. Wayne, Y. W. Teh, and N. Heess. Neural probabilistic motor primitives for humanoid control. In *International Conference on Learning Representations*, 2019.

[53] J. Merel, S. Tunyasuvunakool, A. Ahuja, Y. Tassa, L. Hasenclever, V. Pham, T. Erez, G. Wayne, and N. Heess. Catch & carry: reusable neural controllers for vision-guided whole-body tasks. *ACM Transactions on Graphics (TOG)*, 39(4):39–1, 2020.

[54] A. Escontrela, X. B. Peng, W. Yu, T. Zhang, A. Iscen, K. Goldberg, and P. Abbeel. Adversarial motion priors make good substitutes for complex reward functions. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 25–32. IEEE, 2022.

[55] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/google/jax.

[56] H. van Hasselt, A. Guez, and D. Silver. Deep reinforcement learning with double q-learning. *arXiv e-prints*, 2015.

[57] S. Fujimoto, H. van Hoof, and D. Meger. Addressing function approximation error in actor-critic methods. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmassan, Stockholm, Sweden, July 10-15, 2018*, 2018.

[58] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15:1929–1958, 2014.

[59] J. Ba, J. R. Kiros, and G. E. Hinton. Layer normalization. *ArXiv*, abs/1607.06450, 2016.

[60] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *International Conference on Artificial Intelligence and Statistics*, 2010.

[61] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[62] Y. Tassa, T. Erez, and E. Todorov. Synthesis and stabilization of complex behaviors through online trajectory optimization. *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4906–4913, 2012.